Assignment 2: Programming in NESL

Parallel Functional Programming (PFP) 2017/18

Andrzej Filinski

Due: 2017-12-11

1. Define a (non-trivially parallel) NESL function to extract the constituent words from a string (character vector). For example, the call `words("Hello, world!")` should return `["Hello", "world"]`.

   Hint: Assume first that the words are separated by single spaces, then generalize to any sequence of non-alphabetic characters.

2. Comparing scan algorithms:

   (a) Program the divide-and-conquer *exclusive* scan algorithm.

   (b) Extend the unite-and-conquer scan to arbitrary vector lengths.

   (c) Compare the empirical work and span complexities of the two scans. (It's OK to just consider powers of 2 here.)

3. Define a radix-10 sort function for integer vectors. Investigate its empirical work and span complexity.

4. Comparing string sort algorithms:

   (a) Define a function for comparing strings lexicographically, and use it to define a quicksort funtions for vectors of strings.

   (b) Extend the radix-sort algorithm to work on string vectors.

   (c) Empirically evaluate the performance of the two algorithms.

Please include your code *both* as a `.nesl` source file and as a listing in an appendix of your PDF report. It's fine to put all the requested functions in one file.

Your report should explain (briefly) how the code works, and summarize its empirical performance (in terms of the work/span costs reported by the μNESL interpreter – not actual wall-clock or CPU running times). Note that μNESL includes the special expression form "`time` *exp*", which returns both the value of *exp*, and its work and span costs.